Capitolo 5

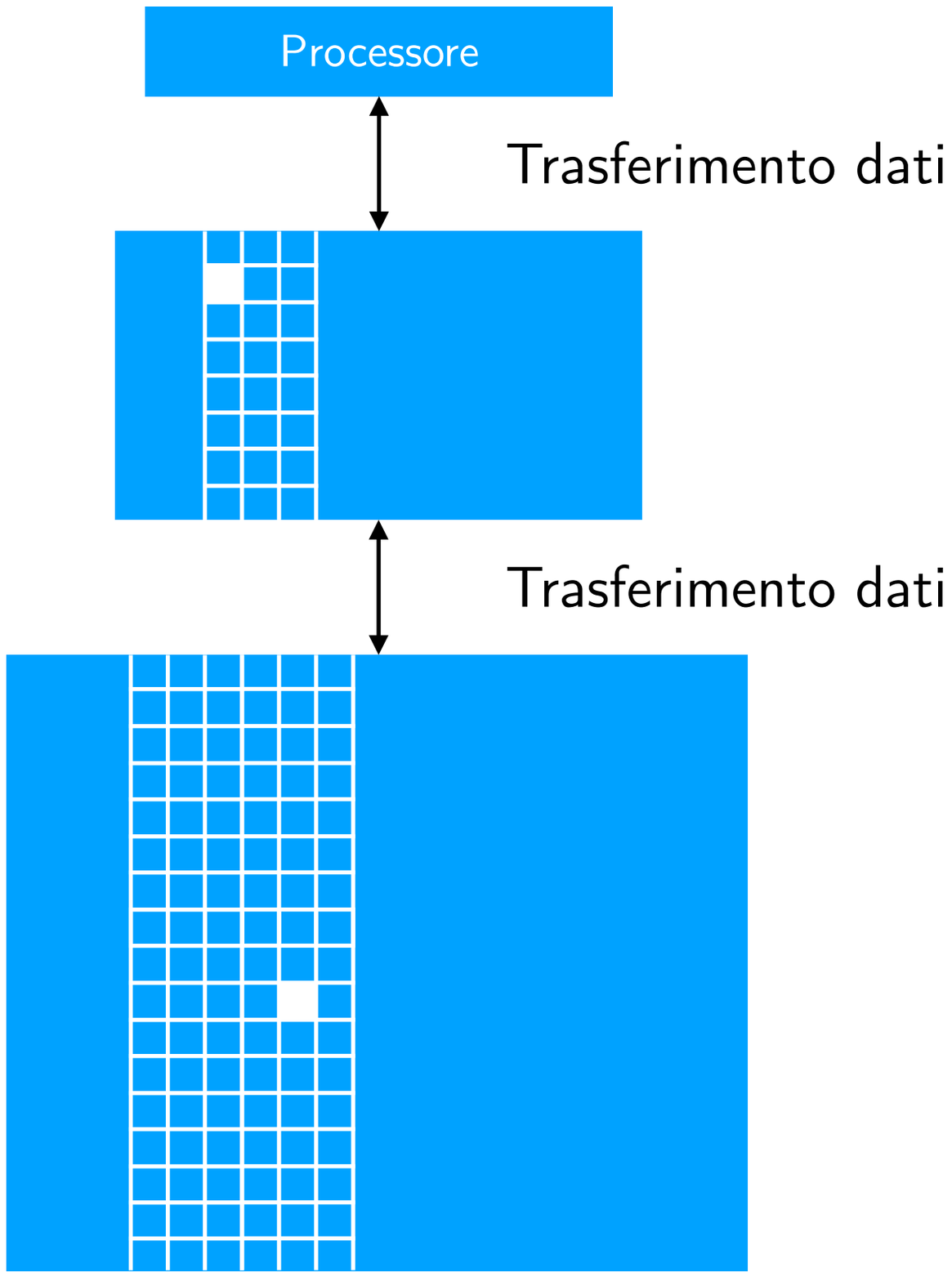
Principio di località

- I programmi accedono a una piccola parte del loro spazio di indirizzamento in ogni istante
- **Località temporale**
  - Oggetti acceduti recentemente saranno con alta probabilità acceduti a breve
  - Per esempio, istruzioni in un ciclo
- **Località spaziale**
  - Oggetti vicini a quelli acceduti recentemente saranno con alta probabilità acceduti a breve
  - Per esempio, accesso sequenziale delle istruzioni, dati negli array

Sfruttare la località

- Gerarchia di memoria
- Memorizzare tutto su disco
- Copiare gli oggetti acceduti di recente (e vicini) dal disco a una memoria DRAM più piccola
  - Memoria principale
- Copiare gli oggetti acceduti ancor più di recente (e ancor più vicini) dalla memoria DRAM a una memoria SRAM più piccola
  - Memoria cache collegata direttamente alla CPU

Livelli di gerarchia di memoria



- **Blocco** (o **linea**): unità di copia
  - Può contenere word multiple
- Se il dato richiesto è presente nel livello superiore
  - **Hit**: accesso soddisfatto dal livello superiore
    - **Hit ratio** = # hit / # accessi
- Se il dato richiesto è assente
  - **Miss**: blocco copiato dal livello inferiore
    - Tempo richiesto: penalità di miss
    - **Miss ratio** = # miss / # accessi = 1 - hit ratio
  - Quindi il dato richiesto è fornito dal livello superiore

Tecnologie della memoria

- RAM Statica (SRAM)
  - 0.5 ns – 2.5 ns, 500$ – 1000$ per GB
- RAM Dinamica (DRAM)
  - 50 ns – 70 ns, 10$ – 20$ per GB
- Memoria flash
  - 5 µs – 50 µs, 0.75$ – 1$ per GB
- Disco magnetico
  - 5 ms – 20 ms, 0.05$ – 0.10 $ per GB
- Memoria ideale
  - Tempo di accesso di una SRAM
  - Capacità e costo/GB di un disco magnetico

Tecnologia SRAM

- Circuiti integrati organizzati come vettori di memoria
  - 6 – 8 transistor per bit
- Stesso tempo di accesso per tutti i dati
  - Ma i tempi di accesso in lettura e scrittura possono essere diversi
- Tempo di accesso molto vicino al ciclo di clock

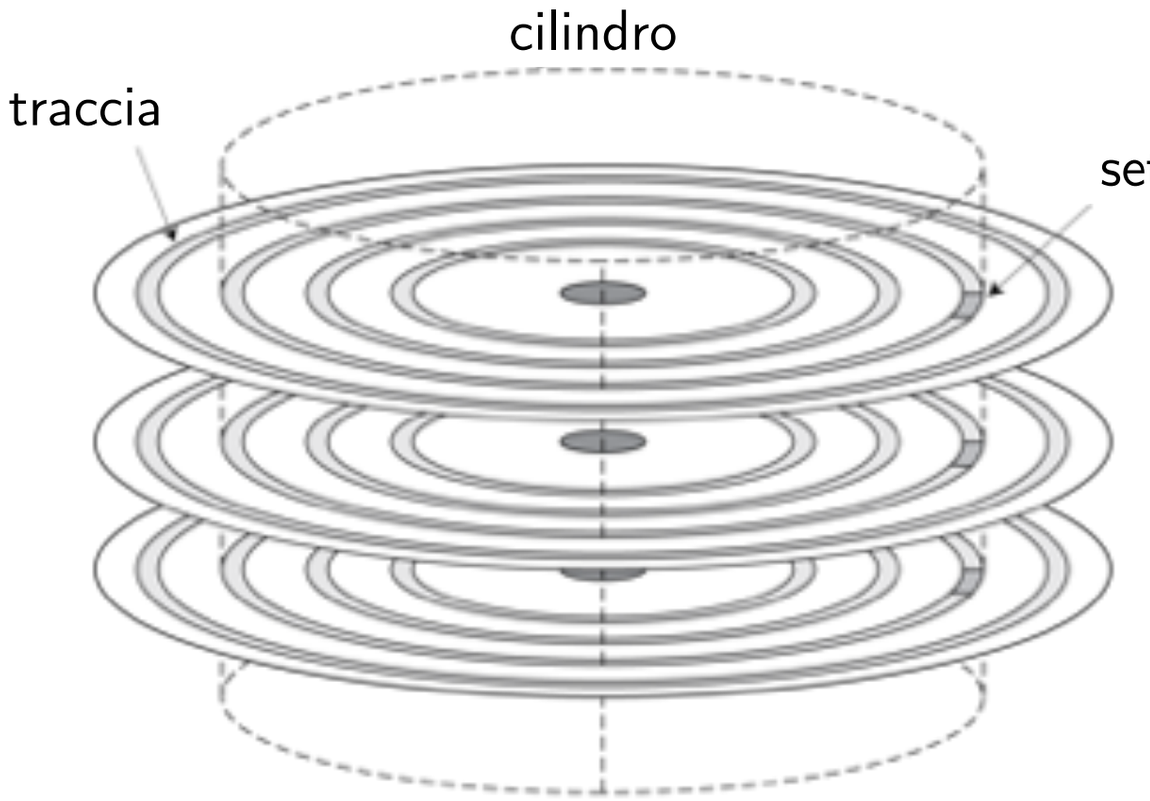
Tecnologia DRAM

- I dati sono memorizzati come carica di un condensatore
  - Un transistor usato per accedere alla carica
  - Deve essere periodicamente aggiorntato
    - Leggere i contenuti e riscriverli
    - Eseguito "a righe"
- I bit di una DRAM sono organizzati come array rettangolari
  - La DRAM accede a una riga intera
  - Modalità burst: fornisce parole successive da una riga con latenza ridotta
- DDR (double data rate) RAM
  - Trasferimento sui fronti in salita e in discesa del clock

Tecnologia flash

- Memoria a sola lettura, cancellabile elettronicamente e programmabile (EEPROM)
  - NOR flash: cella di bit come una porta NOR
    - Accesso in lettura/scrittura casuale
    - Usata per le memoria delle istruzioni nei sistemi embedded
  - NAND flash: cella di bit come una porta NAND
    - Più densa (# bit/area) ma accesso un blocco alla volta
    - Meno costosa per GB
    - Usata per le chiavi USB
- I bit si deteriorano dopo un certo numero (1000+) di scritture
  - Inadatte per rimpiazzare direttamente il disco o la RAM
  - Livellamento dell'usura (*wear leveling*)
    - Rimappare i blocchi di memoria che sono stati scritti più spesso sui blocchi che sono stati scritti meno di frequente
    - Perdita di prestazioni nominali

Tecnologia a disco

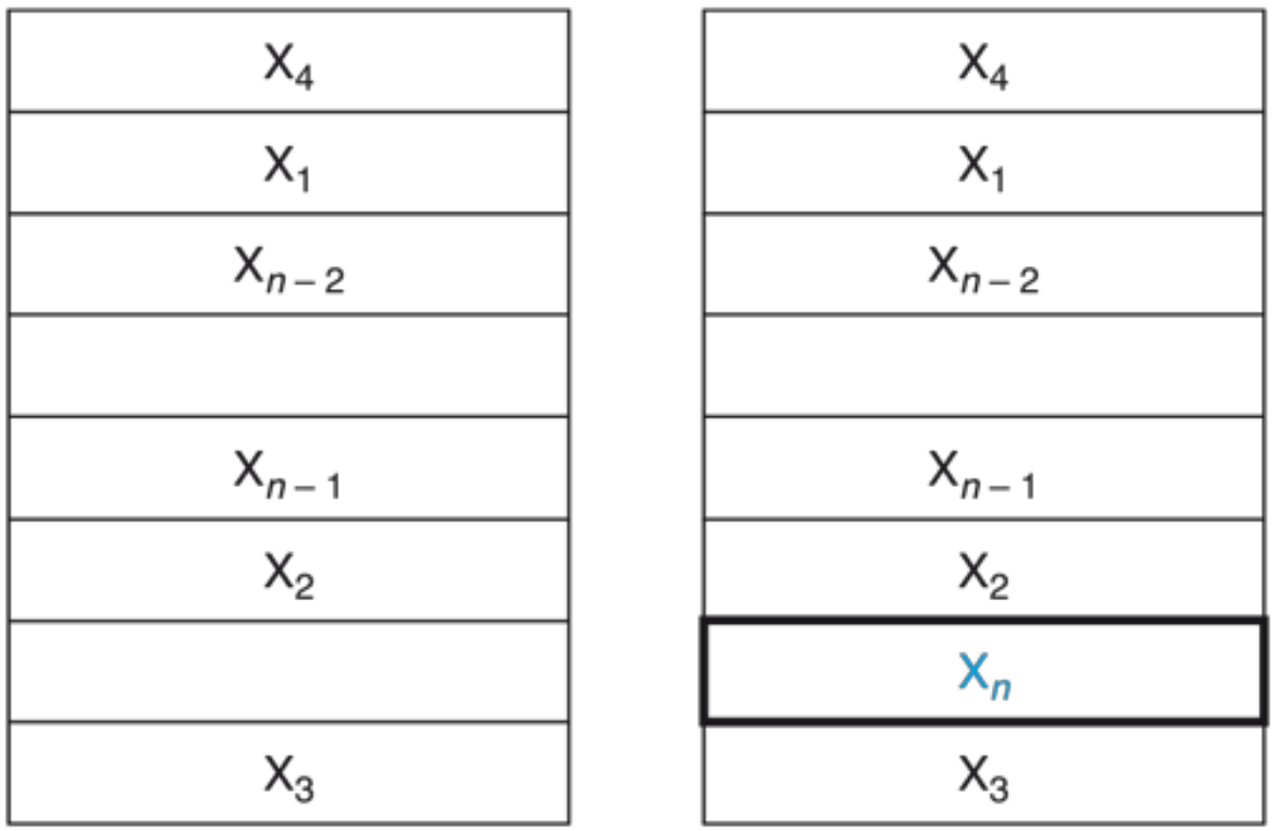


Tecnologia a disco

- Ogni settore memorizza, tra le altre cose
  - ID del settore
  - Dati (512 – 4096 byte)
  - Codice di correzione di errore (ECC)
- L'accesso a un settore richiede
  - Ritardo di attesa se altri accessi sono in attesa
  - Ricerca (*seek*): muovere le testine
  - Latenza rotazionale
  - Trasferimento dati
  - Overhead del controllore

Memoria Cache

- Livello della gerarchia di memoria più vicino alla CPU
- Dati gli accessi X_1, ..., Xn-1, Xn

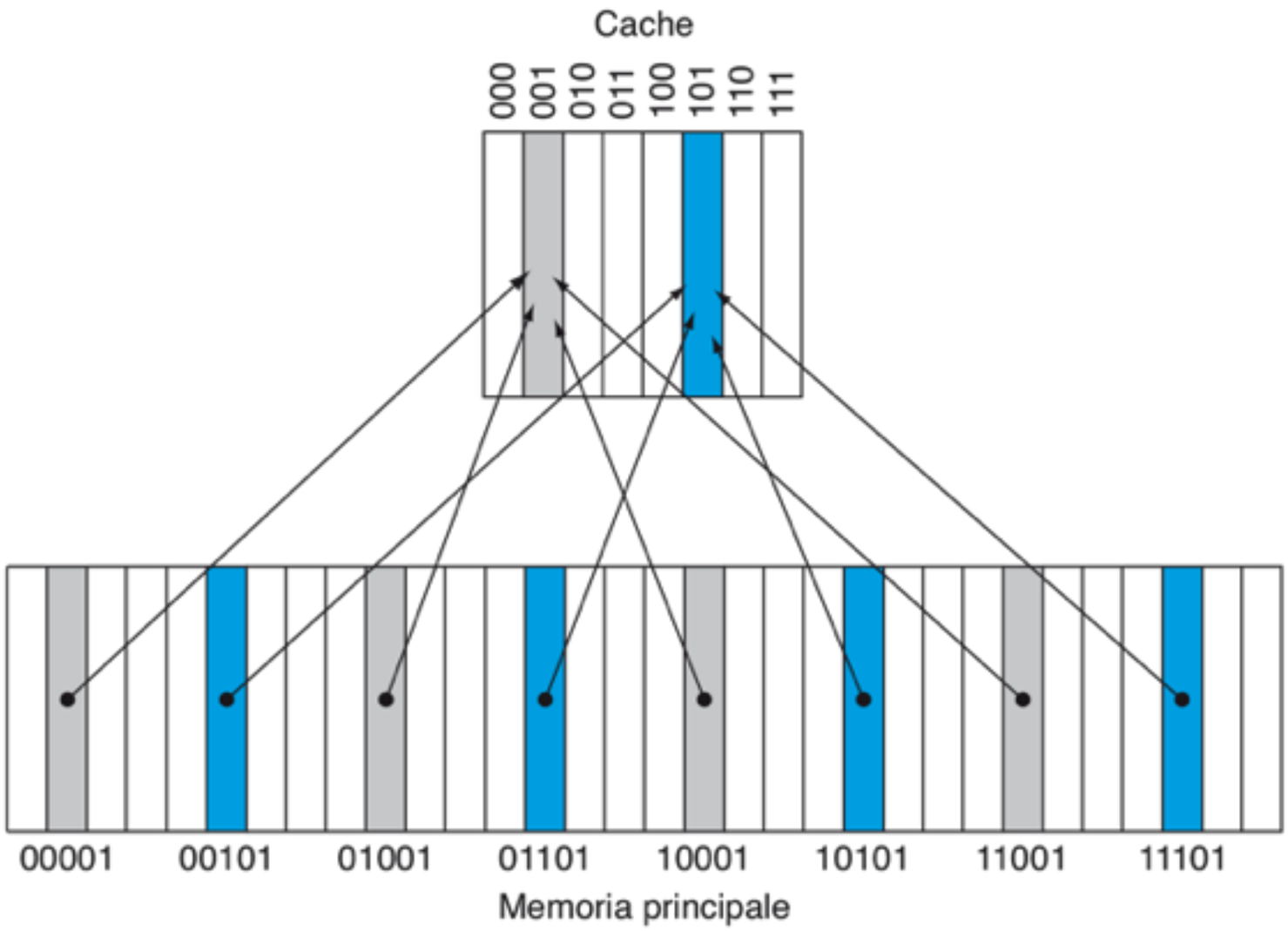


a. Prima di richiedere il dato X_n b. Dopo avere richiesto il dato X_n

- Come sappiamo se il dato è presente in cache?
- Dove andiamo a cercare il dato?

Cache a mappature diretta

- Locazione determinata dall'indirizzo
- **Mappatura diretta**: un'unica possibile scelta
  - (indirizzo del blocco) `modulo` (num. blocchi in cache)



- Num. blocchi è una potenza di 2
- Si usano i bit di indirizzo meno significativi

Tag e bit di validità

- Come sappiamo quale particolare blocco è memorizzato in una data locazione di cache?
  - Si memorizza l'indirizzo del blocco insieme ai dati
  - In realtà, ci servono solo i bit più significativi
  - Costituiscono il **campo tag**
- Cosa succede se non ci sono dati in una locazione di cache?
  - **Bit di validità**: 1 = dato presente, 0 = dato assente
  - Inizialmente posto a 0

Esempio

- 8 blocchi, 1 word per blocco, mappatura diretta
- Stato iniziale

Indice	V	Tag	Dati
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Esempio

Indirizzo di memoria decimale	Indirizzo di memoria binario	Hit/Miss	Blocco di cache
22	10 110	Miss	110

Indice	V	Tag	Dati
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	S	10	Mem[10110]
111	N		

Esempio

Indirizzo di memoria decimale	Indirizzo di memoria binario	Hit/Miss	Blocco di cache
26	11 010	Miss	010

Indice	V	Tag	Dati
000	N		
001	N		
010	S	11	Mem[11010]
011	N		
100	N		
101	N		
110	S	10	Mem[10110]
111	N		

Esempio

Indirizzo di memoria decimale	Indirizzo di memoria binario	Hit/Miss	Blocco di cache
22	10 110	Hit	110
26	11 010	Hit	010

Indice	V	Tag	Dati
000	N		
001	N		
010	S	11	Mem[11010]
011	N		
100	N		
101	N		
110	S	10	Mem[10110]
111	N		

Esempio

Indirizzo di memoria decimale	Indirizzo di memoria binario	Hit/Miss	Blocco di cache
16	10 000	Miss	000
3	00 011	Miss	011
16	10 000	Hit	000

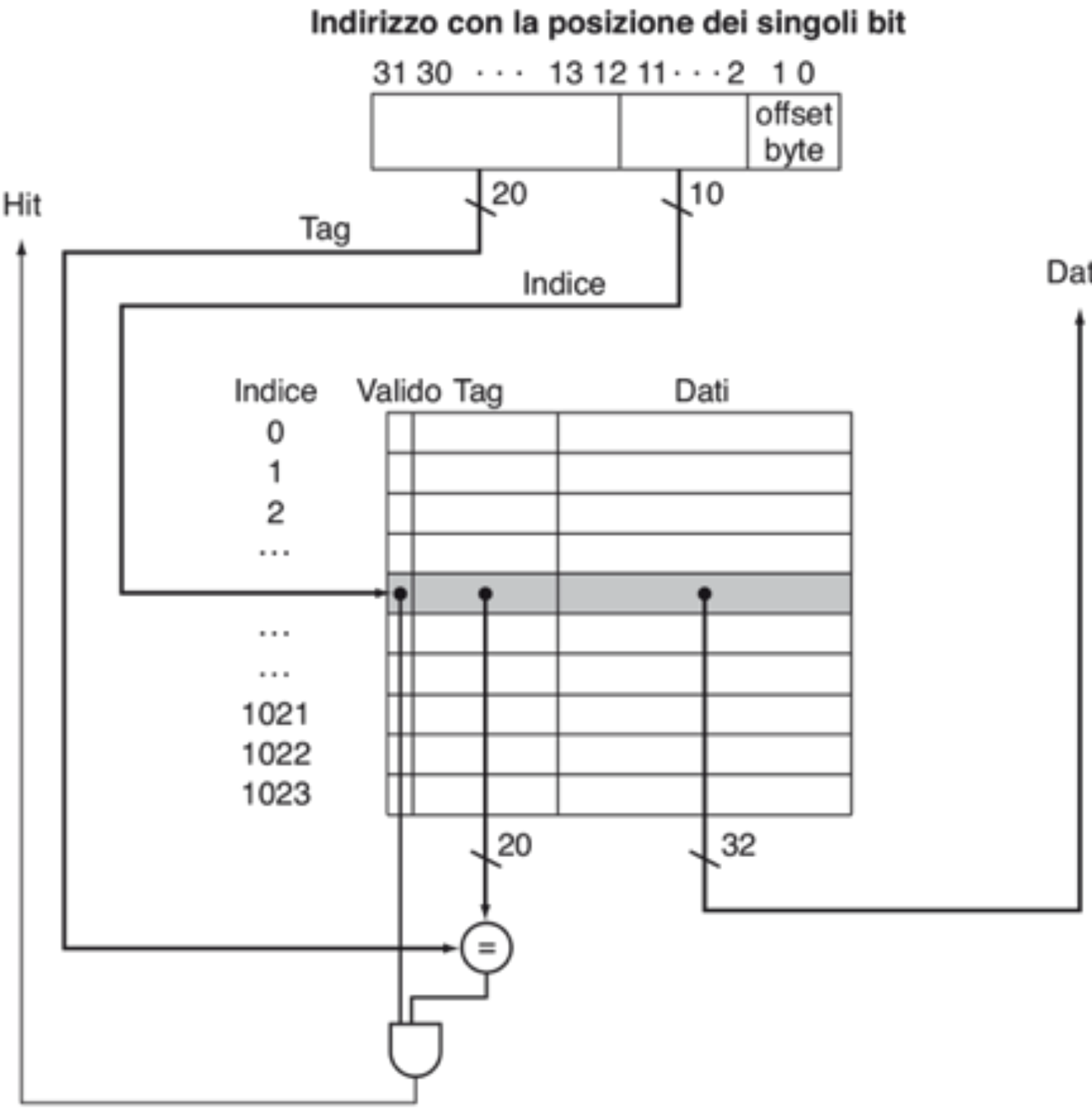
Indice	V	Tag	Dati
000	S	10	Mem[10000]
001	N		
010	S	11	Mem[11010]
011	S	00	Mem[00011]
100	N		
101	N		
110	S	10	Mem[10110]
111	N		

Esempio

Indirizzo di memoria decimale	Indirizzo di memoria binario	Hit/Miss	Blocco di cache
18	10 010	Miss	010

Indice	V	Tag	Dati
000	S	10	Mem[10000]
001	N		
010	S	10	Mem[10010]
011	S	00	Mem[00011]
100	N		
101	N		
110	S	10	Mem[10110]
111	N		

Suddivisione dell'indirizzo



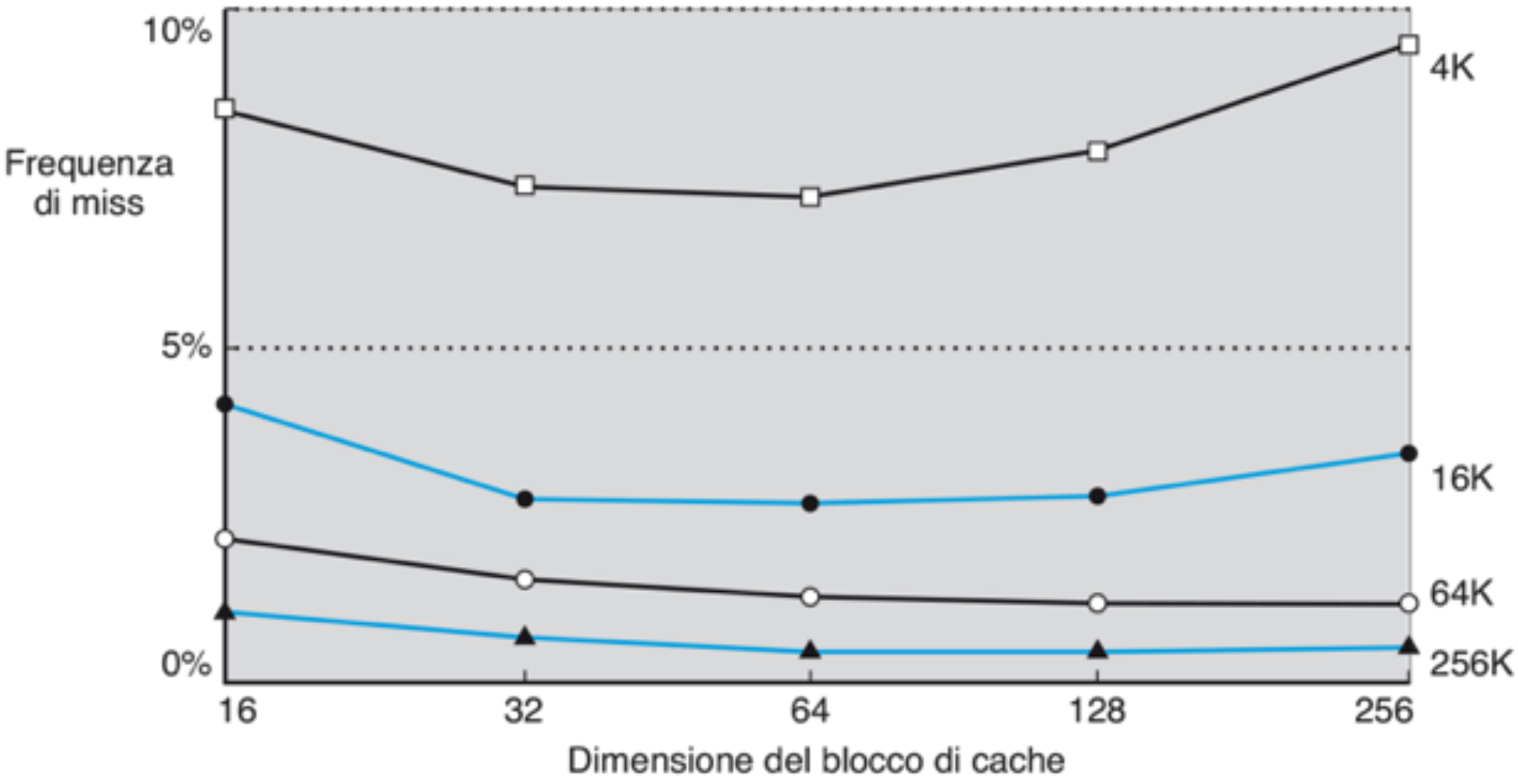
Blocchi composti da più parole

- 64 blocchi, 16 byte per blocco
- A quale numero di blocco corrisponde l'indirizzo 1200?
- Indirizzo di blocco = $\lfloor 1200 \ / \ 16 \rfloor = 75$
- Numero di blocco = 75 `modulo` 64 = 11

Dimensione del blocco

- Blocchi più grandi dovrebbero ridurre il tasso di miss
  - A causa della spazialità locale
- Ma in una cache a dimensione fissa
  - Blocchi più grandi ⇨ Meno blocchi
    - Maggiore competizione ⇨ Aumento del tasso di miss
  - Blocchi più grandi ⇨ Maggior "sporcizia"
- Maggiore penalità di miss
  - Può vanificare i benefici di un tasso di miss ridotto
  - Tecniche avanzate per aiutare in questi casi

Prestazioni



Miss della cache

- In caso di hit, la CPU procede normalmente
- In caso di miss
  - La pipeline della CPU entra in stallo
  - L'unità di controllo preleva il blocco dal livello successivo della gerarchia di memoria
- Miss della cache per un'istruzione
  - Si ripete dalla fase di fetch l'istruzione
- Miss della cache per un dato
  - Si completa l'accesso al dato

Write-Through

- Nel caso di hit in scrittura, si potrebbe semplicemente aggiornare il blocco di cache
  - Ma in tal caso cache e memoria sarebbero inconsistenti
- **Write through**: aggiornare anche la memoria
- Ma rende le scritture più lente
  - Ad esempio, se il CPI di base è 1, il 10% delle istruzioni sono di scrittura e la scrittura richiede 100 cicli di clock
    - CPI effettivo = 1 + 0.1 x 100 = 11
- Soluzione: **buffer di scrittura**
  - Contiene i dati in attesa di essere scritti in memoria
  - La CPU continua immediatamente
    - Entra in stallo sulla scrittura solo se il buffer di scrittura è pieno

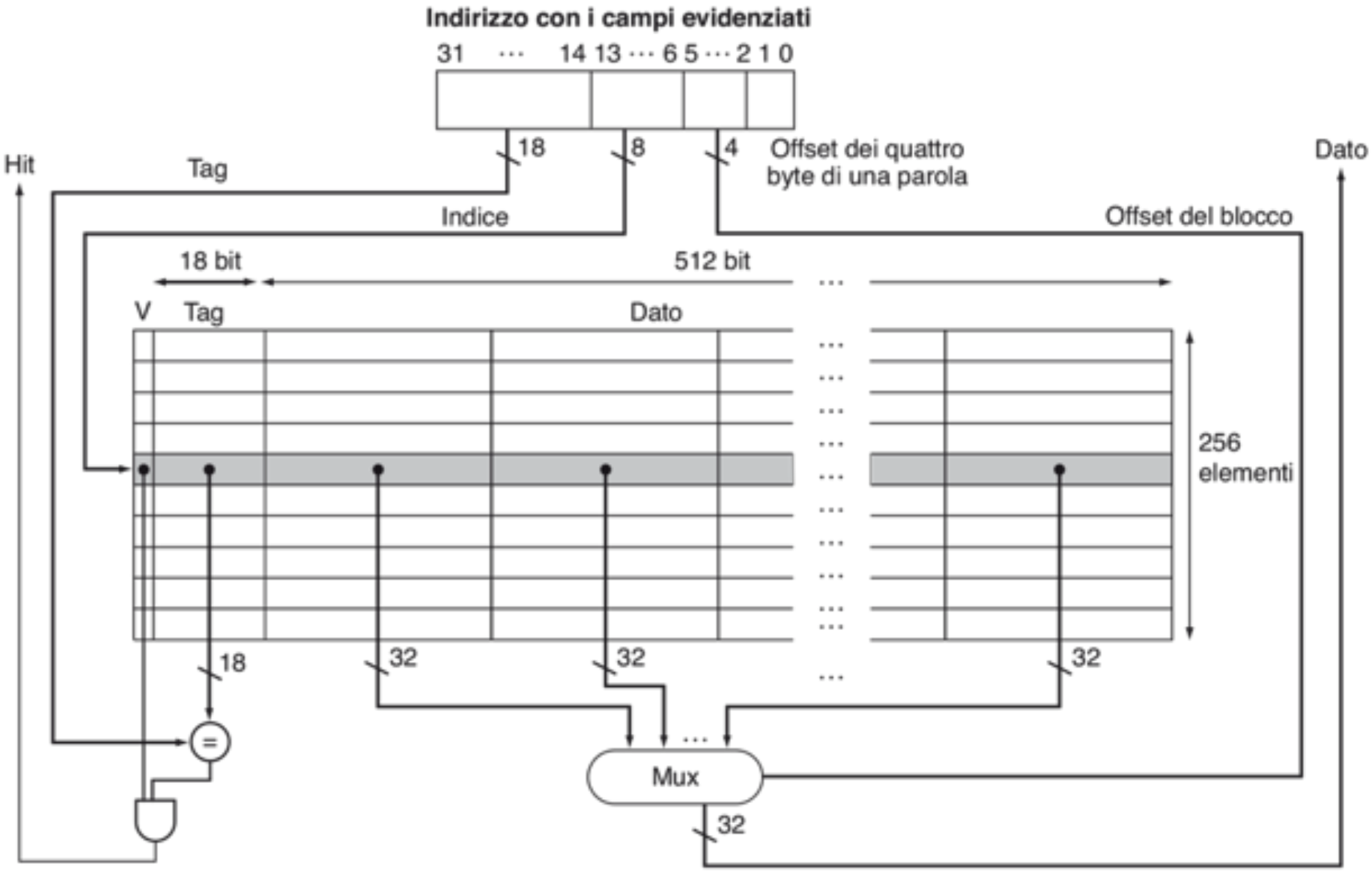
Write-Back

- Alternativa: nel caso di hit in scrittura si aggiorna solo il blocco di cache
  - Bisogna tenere traccia di quali blocchi sono "sporchi" (*dirty*)
- Quando un blocco "sporco" è rimpiazzato
  - Si ri-scrive sulla memoria
  - Si può usare un buffer di scrittura per permettere di rimpiazzare il blocco che deve essere letto per primo

Esempio: Intrinsity FastMATH

- Processore MIPS dedicato
  - Pipeline a 12 stadi
  - Accesso a dati e istruzioni in ciascun ciclo
- Split cache: cache separate per istruzioni e dati
  - Ciascuna di 16 KB: 256 blocchi per 16 parole/blocco
  - Cache dati: write-through o write-back
- Tassi di miss:
  - Cache istruzioni: 0.4 %
  - Cache dati: 11.4 %
  - Media pesata: 3.2 %

Esempio: Intrinsity FastMATH



Misurare le prestazioni della cache

- Componenti del tempo di CPU
  - Cicli di esecuzione del programma
    - Incluso il tempo di hit in cache
  - Cicli di stallo della memoria
    - Principalmente dovuti alle miss in cache
- Con assunzioni semplificative

Cicli di stallo della memoria
= Num. accessi in memoria x tasso di miss x penalità di miss
= Num. istruzioni x num miss/istruzione x penalità di miss

Esempio

- Dati
  - Tasso di miss della I-cache = 2 %
  - Tasso di miss della D-cache = 4 %
  - Penalità di miss = 100 cicli
  - CPI base (cache ideale) = 2
  - Load e store sono il 36% delle istruzioni
- Cicli di miss per istruzione
  - I-cache: 0.02 x 100 = 2
  - D-cache: 0.36 x 0.04 x 100 = 1.44
- CPI reale = 2 + 2 + 1.44 = 5.44
  - La CPU ideale è 5.44/2 = 2.72 volte più veloce

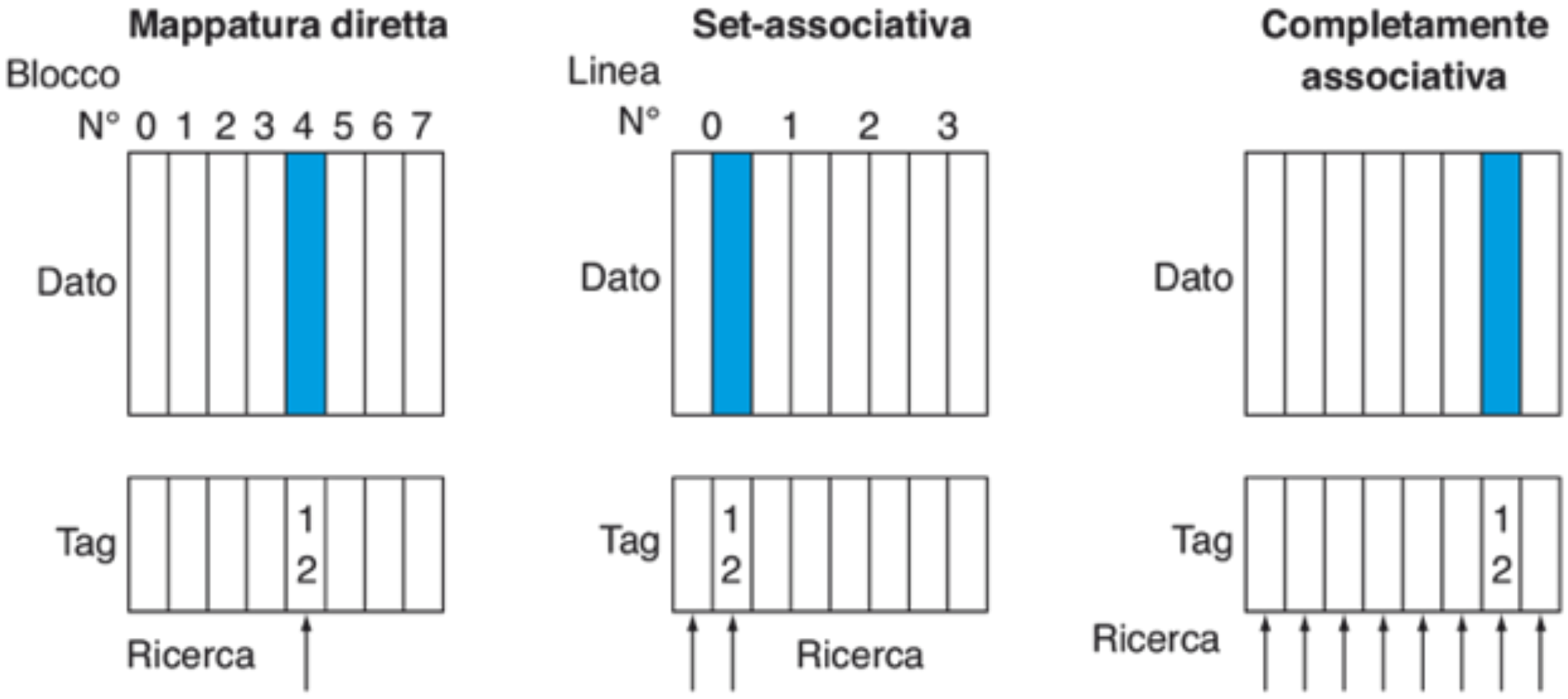
Sommario sulle prestazioni

- Quando le prestazioni della CPU aumentano
  - La penalità di miss diventa più significativa
- Diminuendo il CPI base
  - Una maggior parte del tempo è speso per gli stalli di memoria
- Aumentando la frequenza di clock
  - Gli stalli di memoria costano più cicli di clock
- Non si può trascurare il comportamento della cache quando si valutano le prestazioni di un sistema

Cache associative

- **Completamente associative**
  - Ogni blocco di memoria può andare in qualsiasi blocco di cache
  - Richiede di ricercare in parallelo tutti i blocchi di cache
  - Un comparatore per blocco di cache (costoso)
- **Set-associative**
  - Ogni set contiene *n* blocchi di cache
  - Il numero di blocco determina il set
    - (Numero di blocco) `modulo` (numero di set nella cache)
  - Richiede di ricercare in parallelo tutti i blocchi di un set
  - Solo n comparatori (meno costoso)

Esempio di cache associativa



Varianti di associatività

Cache set-associativa a una via
(a mappatura diretta)

Blocco	Tag	Dato
0		
1		
2		
3		
4		
5		
6		
7		

Cache set-associativa a due vie

Linea	Tag	Dato	Tag	Dato
0				
1				
2				
3				

Cache set-associativa a quattro vie

Linea	Tag	Dato	Tag	Dato	Tag	Dato	Tag	Dato
0								
1								

Cache set-associativa a otto vie (completamente associativa)

Tag	Dato	Tag	Dato	Tag	Dato	Tag	Dato	Tag	Dato	Tag	Dato	Tag	Dato	Tag	Dato

Esempio

- Confrontiamo una cache a 4 blocchi
  - A mappatura diretta, set-associativa a 2 vie, completamente associativa
- Sequenza di accesso ai blocchi: 0, 8, 0, 6, 8

- A mappatura diretta

Indirizzo di blocco	Indice di cache	Hit/miss	Contenuto della cache dopo l'accesso			
			0	1	2	3
0	0	miss	Mem[0]			
8	0	miss	Mem[8]			
0	0	miss	Mem[0]			
6	2	miss	Mem[0]		Mem[6]	
8	0	miss	Mem[8]		Mem[6]	

Esempio

- Set-associativa a 2 vie

Indirizzo di blocco	Indice di cache	Hit/miss	Contenuto della cache dopo l'accesso			
			Set 0		Set 1	
0	0	miss	Mem[0]			
8	0	miss	Mem[0]	Mem[8]		
0	0	hit	Mem[0]	Mem[8]		
6	0	miss	Mem[0]	Mem[6]		
8	0	miss	Mem[0]	Mem[6]		

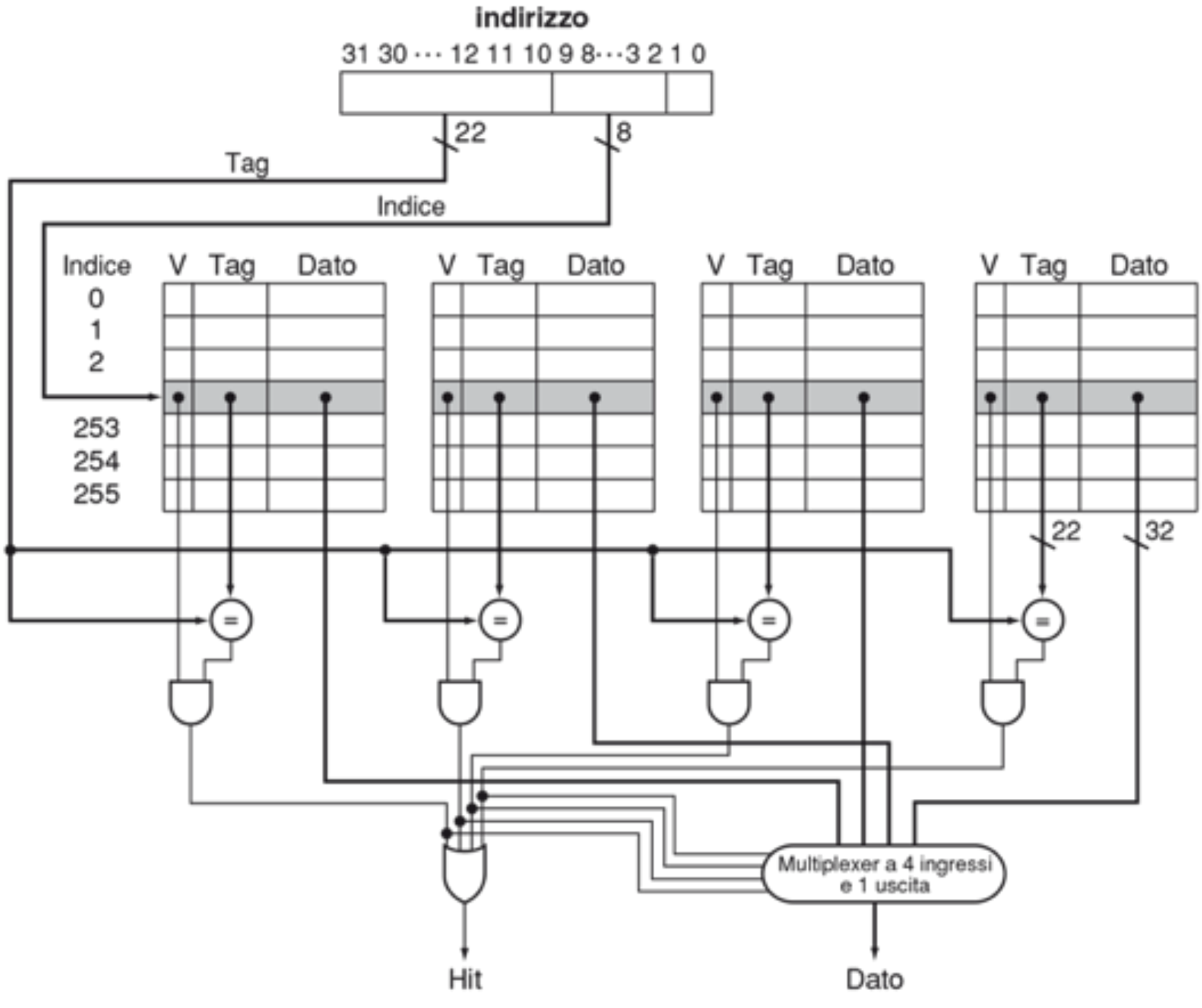
- A mappatura diretta

Indirizzo di blocco		Hit/miss	Contenuto della cache dopo l'accesso			
0		miss	Mem[0]			
8		miss	Mem[0]	Mem[8]		
0		hit	Mem[0]	Mem[8]		
6		miss	Mem[0]	Mem[8]	Mem[6]	
8		hit	Mem[0]	Mem[8]	Mem[6]	

Quanta associatività

- Aumentare l'associatività diminuisce il tasso di miss
  - Ma con benefici sempre minori
- Simulazione di un sistema con 64 KB D-cache, blocchi di 16 parole:
  - 1 via: 10.3%
  - 2 vie: 8.6%
  - 4 vie: 8.3%
  - 8 vie: 8.1%

Organizzazione di una cache set-associativa



Politica di rimpiazzamento

- A mappature diretta: nessuna scelta
- Set-associativa
  - Preferire blocchi non validi, se possibile
  - Altrimenti, scegliere tra i rimanenti blocchi nell'insieme
- **Utilizzo meno recente** (LRU, *least recently used*)
  - Scegliere il blocco rimasto inutilizzato più a lungo
    - Semplice per 2 vie, gestibile per 4 vie, troppo complesso per più vie
- **Casuale** (*random*)
  - Approssimativamente esibisce le stesse prestazioni di LRU per elevata associatività

Cache multilivello

- Cache principale (di primo livello, L1) unita alla CPU
  - Piccola ma veloce
- Cache di secondo livello (L2) serve le miss della cache principale
  - Più grande, più lenta, ma più veloce della memoria principale
- La memoria principale serve le miss della cache L2
- Alcuni sistemi ad elevate prestazioni includono una cache L3